Plexon® Inc

Reading PL2™ Files With C++

4/19/2013

Revision History (most recent last):			
DATE	REVISION	DESCRIPTION	ORIGINATOR
4/19/2013	-	Initial Version	A. Kirillov

Plexon Inc **Tel: 214 369 4957**
6500 Greenville Ave. # 700 LB33 **Fax: 214 369 1775**
Dallas TX 75206 **www.plexon.com**

Reading PL2 Files With C++

This package provides the information and the libraries required to read Plexon PL2 files using C++ on Windows platforms. A PL2 file is a Plexon data file containing action-potential (spike) timestamps and waveforms (spike channels), event timestamps and event values (event channels), and continuous variable data (continuous A/D channels).
This package contains the following:

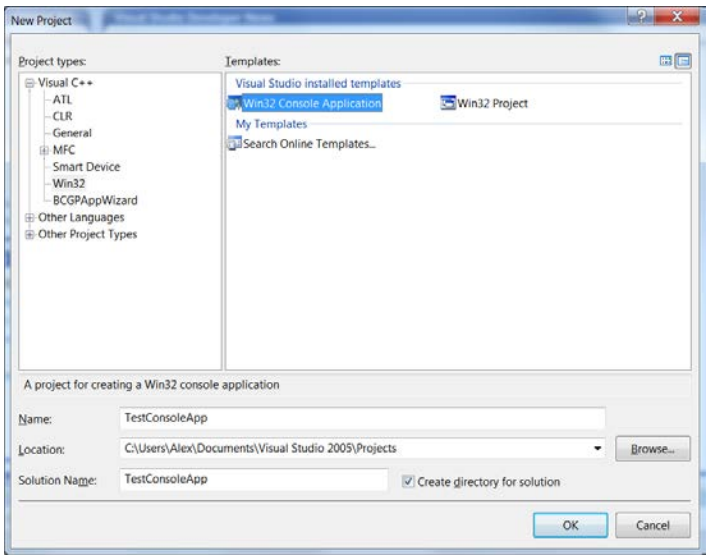
- **Bin** folder containing 32-bit and 64-bit builds of the library *PL2FileReader.dll* and the test program *PL2FileReaderTest.exe*
- **Win32SampleCode** folder containing:
  - Header files describing functions available in *PL2FileReader.dll*:
    - *PL2FileStructures.h* - header file that contains definitions of various structures used in the PL2 file access functions
    - *PL2FileReader.h* - header file that contains documentation on all the functions available in PL2FileReader.dll library
  - Files of a Visual Studio 2005 console project Pl2FileReaderTest demonstrating how to use various functions in PL2FileReader.dll:
    - *PL2FileReaderTest.cpp* – source code file. The code reads spike, event and continuous data from a PL2 file and prints a sample of data of each type
    - *PL2FileReaderTest.sln* – Visual Studio solution file. Open this file in Visual Studio 2005 or later to compile the project
    - *PL2FileReader.lib* – lib file for PL2FileReader.dll. Link this .lib file into an executable that uses PL2FileReader.dll
    - *stdafx.h, stdafx.cpp* – standard precompiled header files for a Visual Studio console project
    - *PL2FileReaderTest.exe* – compiled test program. To run the program, open command line window and type:

      PL2FileReaderTest <pl2_file_path>

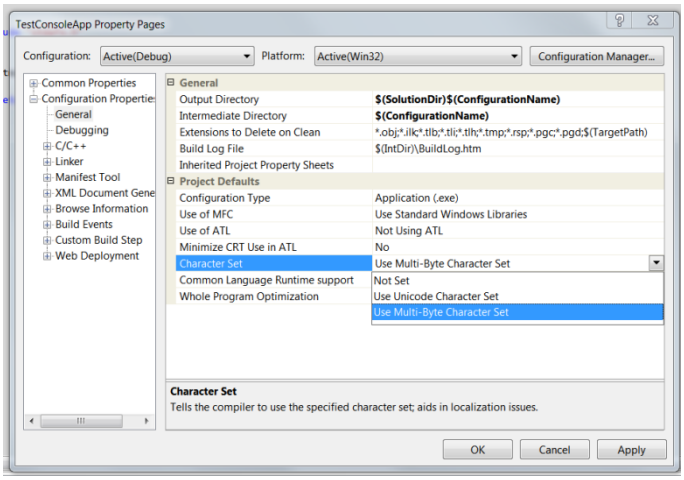
If you have any questions about reading PL2 files, please feel free to contact us at support@plexon.com

Tutorial: How to use PL2FileReader.dll in C++ code

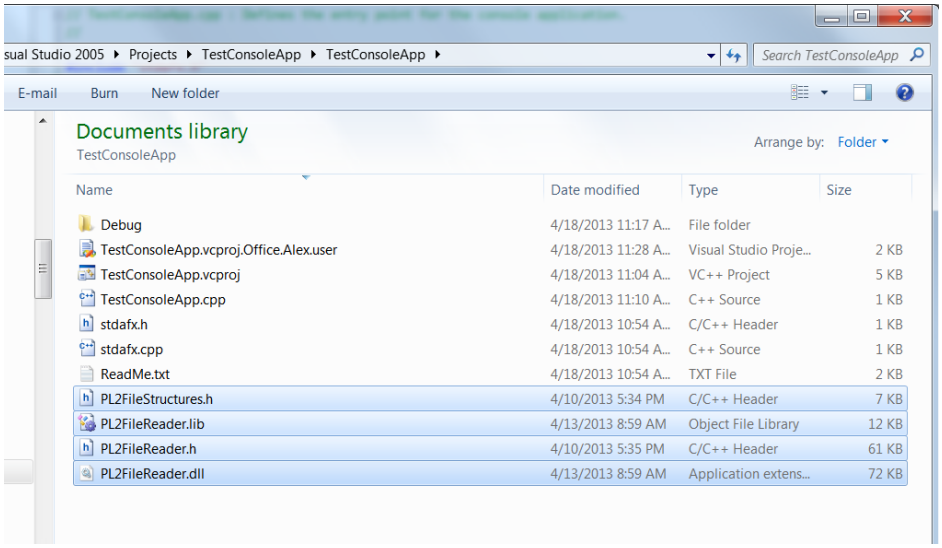
1. Create a console C++ project in Visual Studio:



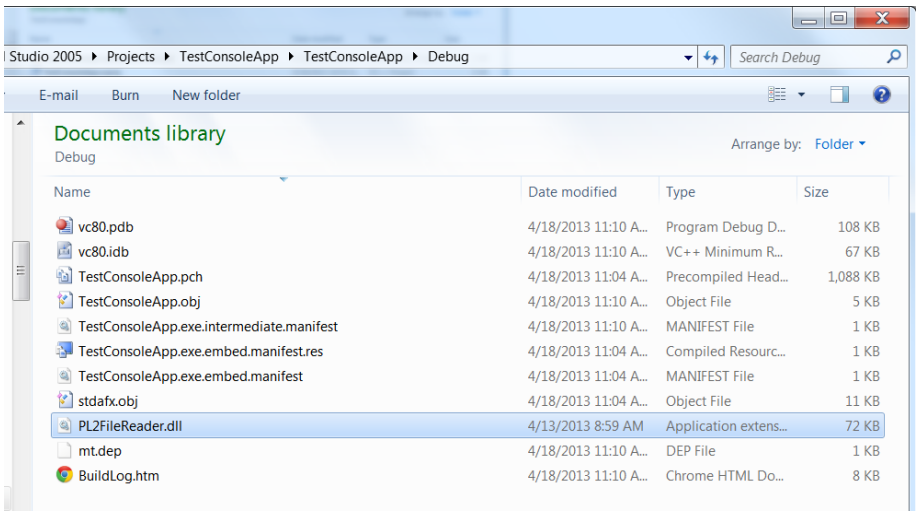
2. Select Project | Properties menu command and specify 'Use Multi-byte Character Set':



3. Copy *PL2FileStructures.h, PL2FileReader.h, PL2FileReader.lib* and *PL2FileReader.dll* to the project source code folder.



4. Copy PL2FileREader.dll to the project's Debug folder:



5. Include PL2FileReader.h header file and specify that the linker should include PL2FileReader.lib:

```
#include "stdafx.h"
#include "PL2FileReader.h"

#pragma comment(lib, "PL2FileReader")

int _tmain(int argc, _TCHAR* argv[])
{
        return 0;
}
```


6. Add code to open a PL2 file, read file information and print channel counts:

```
#include "stdafx.h"
#include "PL2FileReader.h"

#pragma comment(lib, "PL2FileReader")

int _tmain( int argc, _TCHAR* argv[] )
{
    int fileHandle = 0;
    PL2_OpenFile( "C:\\PlexonData\\test.pl2", &fileHandle );
    PL2FileInfo fileInfo;
    PL2_GetFileInfo( fileHandle, &fileInfo );
    printf( "Spike channels: %d\n", fileInfo.m_TotalNumberOfSpikeChannels );
    printf( "Analog channels: %d\n", fileInfo.m_TotalNumberOfAnalogChannels );
    printf( "Digital channels: %d\n", fileInfo.m_NumberOfDigitalChannels );
    PL2_CloseFile( fileHandle );
    return 0;
}
```

7. Compile and run the program.

Functions

Here is the list of functions available in PL2FileReader.dll:

PL2_OpenFile
PL2_CloseFile
PL2_CloseAllFiles
PL2_GetLastError
PL2_GetFileInfo
PL2_GetAnalogChannelInfo
PL2_GetAnalogChannelInfoByName
PL2_GetAnalogChannelInfoBySource
PL2_GetAnalogChannelData
PL2_GetAnalogChannelDataByName
PL2_GetAnalogChannelDataBySource
PL2_GetSpikeChannelInfo
PL2_GetSpikeChannelInfoByName
PL2_GetSpikeChannelInfoBySource
PL2_GetSpikeChannelData
PL2_GetSpikeChannelDataByName
PL2_GetSpikeChannelDataBySource
PL2_GetDigitalChannelInfo
PL2_GetDigitalChannelInfoByName
PL2_GetDigitalChannelInfoBySource
PL2_GetDigitalChannelData
PL2_GetDigitalChannelDataByName
PL2_GetDigitalChannelDataBySource
PL2_GetStartStopChannelInfo
PL2_GetStartStopChannelData
PL2_ReadFirstDataBlock

PL2_ReadNextDataBlock
PL2_GetDataBlockInfo
PL2_GetSpikeDataBlockTimestamps
PL2_GetSpikeDataBlockUnits
PL2_GetSpikeDataBlockWaveforms
PL2_GetAnalogDataBlockTimestamp
PL2_GetAnalogDataBlockValues
PL2_GetDigitalDataBlockTimestamps
PL2_GetDigitalDataBlockValues
PL2_GetStartStopDataBlockTimestamps
PL2_GetStartStopDataBlockValues

Please note that PL2FileReader.h contains documentation for each function. Documentation for each function describes all the parameters of the function and includes sample source code. For example, here is the documentation for PL2_GetAnalogChannelDataByName:

```
/*------- PL2_GetAnalogChannelDataByName ----------------------
    Purpose:
        Retrieve analog channel data

        Analog data come in fragments. Each fragment has a timestamp
        and a number of a/d data points. The timestamp corresponds to
        the time of recording of the first a/d value in this fragment.

    Parameters:
    int fileHandle - file handle
    const char* channelName - analog channel name
    unsigned long long* numFragmentsReturned - pointer to number of fragments
    unsigned long long* numDataPointsReturned - pointer to number of data points
    long long* fragmentTimestamps - pointer to an array of fragment timestamps
                the array should have at least PL2AnalogChannelInfo.m_MaximumNumberOfFragments
elements
                The timestamps are returned in ticks. To convert timestamps to seconds, divide by
                            PL2FileInfo.m_TimestampFrequency

    long long* fragmentCounts - pointer to an array of fragment counts
                the array should have at least PL2AnalogChannelInfo.m_MaximumNumberOfFragments
elements


    short* values - pointer to an array of raw a/d values
                            the array should have at least PL2AnalogChannelInfo.m_NumberOfValues
elements

              To convert raw values to Volts, multiply by
PL2AnalogChannelInfo.m_CoeffToConvertToUnits

    Return Values:
        1 - function succeeded
        0 - function failed (use PL2_GetLastError() to retrieve error description)

    Sample Code:

    int fileHandle = 0;
    PL2_OpenFile( "C:\\PlexonData\\test.pl2", &fileHandle );
    PL2FileInfo fileInfo;
    PL2_GetFileInfo( fileHandle, &fileInfo );
    // get data for analog channel "FP01"
    PL2AnalogChannelInfo channelInfo;
    PL2_GetAnalogChannelInfoByName( fileHandle, "FP01", &channelInfo );
    if ( channelInfo.m_NumberOfValues > 0 ) {
        unsigned long long numFragmentsReturned = 0;
```

```
        unsigned long long numDataPointsReturned = 0;
        long long* fragmentTimestamps = new long long[( size_t
)channelInfo.m_MaximumNumberOfFragments ];
        unsigned long long* fragmentCounts = new unsigned long long[( size_t
)channelInfo.m_MaximumNumberOfFragments ];
        short* values = new short[( size_t )channelInfo.m_NumberOfValues];

        PL2_GetAnalogChannelDataByName( fileHandle, "FP01", &numFragmentsReturned,
&numDataPointsReturned
        , fragmentTimestamps, fragmentCounts, values );

        // print first few timestamps and values
        // if the first fragment count is more than 4 data points
        if ( numDataPointsReturned >= 4 && fragmentCounts[0] >= 4 ) {
            printf( "Timestamp (sec)   Value (mV)\n" );
            double step = 1.0 / channelInfo.m_SamplesPerSecond;
            double fragmentTimestampInSeconds = fragmentTimestamps[0] /
fileInfo.m_TimestampFrequency;
            for ( size_t valueIndex = 0; valueIndex < 4; ++valueIndex ) {
                double dataPointTimestampInSeconds = fragmentTimestampInSeconds + step *
valueIndex;
                double valueInMilliVolts = values[valueIndex]*channelInfo.m_CoeffToConvertToUnits
* 1000;
                printf( "%15.6f %12.6f\n", dataPointTimestampInSeconds, valueInMilliVolts );
            }
        }
        delete []fragmentTimestamps;
        delete []fragmentCounts;
        delete []values;
    }
    */
```

Error Handling

Most of the functions in the SDK return 1 if function succeeded and return 0 if function failed. To get the error information, use PL2_GetLastError function. For example:

```
    PL2FileInfo info;
    if ( !PL2_GetFileInfo( fileHandle, &info ) ) {
        char error[1024];
        PL2_GetLastError( error, 1024 );
        cout << "unable to get file info: " << error << endl;
    }
```

For more code examples, please refer to the code in PL2FileReaderTest.cpp.

PL2FileReaderTest Program

A compiled PL2 file reader test program is provided in the SDK. To use the program,

- Open a command line prompt in the **Win32SampleCode** folder of the SDK
- To print information about the file C:\PlexonData\test.pl2, type:

```
PL2FileReaderTest "C\PlexonData\test.pl2"
```

Here is an example of the program's output